0-ITM

0-in-the-middle
15 € Malware Lab

HELP → ABOUT

- Real job: security engineer and analyst
- Hobby: everything technological related, with a special interest in security stuff
- I worked on a lot of open source projects, software and hardware related. I built a custom Arduino compatible board as a programmable GPS tracker (hereyouARE)
- Find details at: http://enerduino.blogspot.com
- I also developed the first PoC af an Arduino code injection, published at https://github.com/cecio

THE NEED

- Perform "quick and dirty" analysis on the fly
- Avoid to setup complex malware labs
- Reduce the turn-around time and be faster in providing details about what is going on

….. and yes, you are right, I'm lazy….

THE NEED

Why not using the malware lab that someone else (the end-user), already prepared for us?

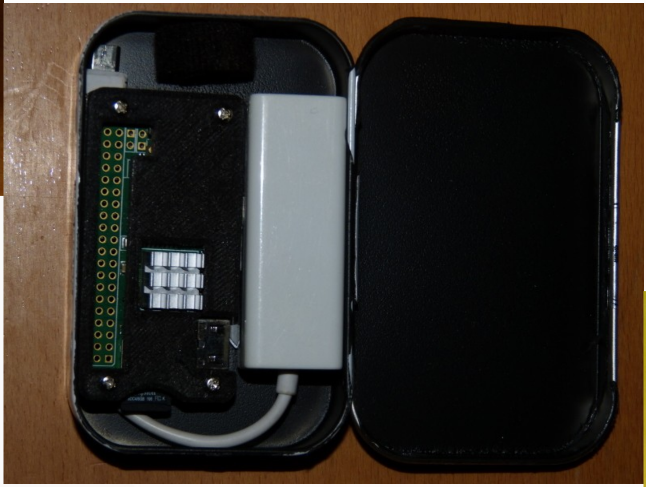
We just need our favorites tools and a safe way to interact with the system.

MY SOLUTION

A cheap, portable, customizable, quick to configure (if not already configured) external device, to proceed with our analysis, without compromising the existing production (or safe) environment.

MY SOLUTION

The Original Celebrated
ALTOIDS
CURIOUSLY STRONG MINTS
With Other Natural Flavors
PEPPERMINT
NET WT 1.76 OZ (50g)



0-ITM

0-ITM is a portable (ALTOIDS contained), cheap, easy-to-setup malware analysis tool, built to speed up the reverse engineering and analysis of malware or any other software. Based on Raspberry Pi 0 W.

Features:

+ analysis of infected PC where it's not obvious how to extract the malware

+ analysis of malware requiring a lot of human interaction

+ analysis of malware with strong anti VM functionalities

+ analysis, through the Wifi, of Android, iOS and in general IoT behaviors

+ highly portable and configurable

+ ready to use in a quick way (menu driven)

+ veeeery cheap

+ several operating mode: AP, station, sniffing mode

+ secure (internal net are not routed, in routed mode)

It has been specifically built for Android base software: it contains a customized version of fakenet-ng to handle Android requests (pending a pull request on github).

O-ITM

As an example, some of the software installed and configured on the platform is

- inetsim
- fakenet (custom version to analyize Android app)
- honeyd
- Dionaea
- IRC server (for old school malware)
- yara rules
- balbuzard, floss
- tcpdump, ettercap, tshark, nmap, mitmproxy, irc, etc
- radare2, pyew
- ...and many other...

Part of the setup, is an ethernet adapter, so you can choose on which interface you can conduct your investigations: wired or wifi.

O-ITM

We can choose the wired or the Wifi interface, and then intercept all the traffic going through:

```
iptables -t nat -A PREROUTING -i eth0 -j REDIRECT

iptables -t nat -A PREROUTING -i wlan0 -j REDIRECT
```

O-ITM

The configurations will be menu-driven, and the core of the project is to be quick and not loosing time in configuring the environment:

This is a sample of the menu available:

[1] - Set time and date
[2] - InetSim
[3] - FakeNet-NG
[4] - Dionaea
[5] - HoneyD
[6] - IRC Server
[7] - mitmproxy transparent (requires Station/Routed mode)
[8] - Switch AP <-> Station mode
[9] - Sniff Wifi
[10] - Update Software

0-ITM

General infos:

Default Password:	Zeroitm (change it, please!)
Default SSID:	0itm
Default SSID password:	0itm0nly (change it, please!)
IP wifi:	192.168.10.1
IP wired:	192.168.0.1
SSH Listening on port:	22222

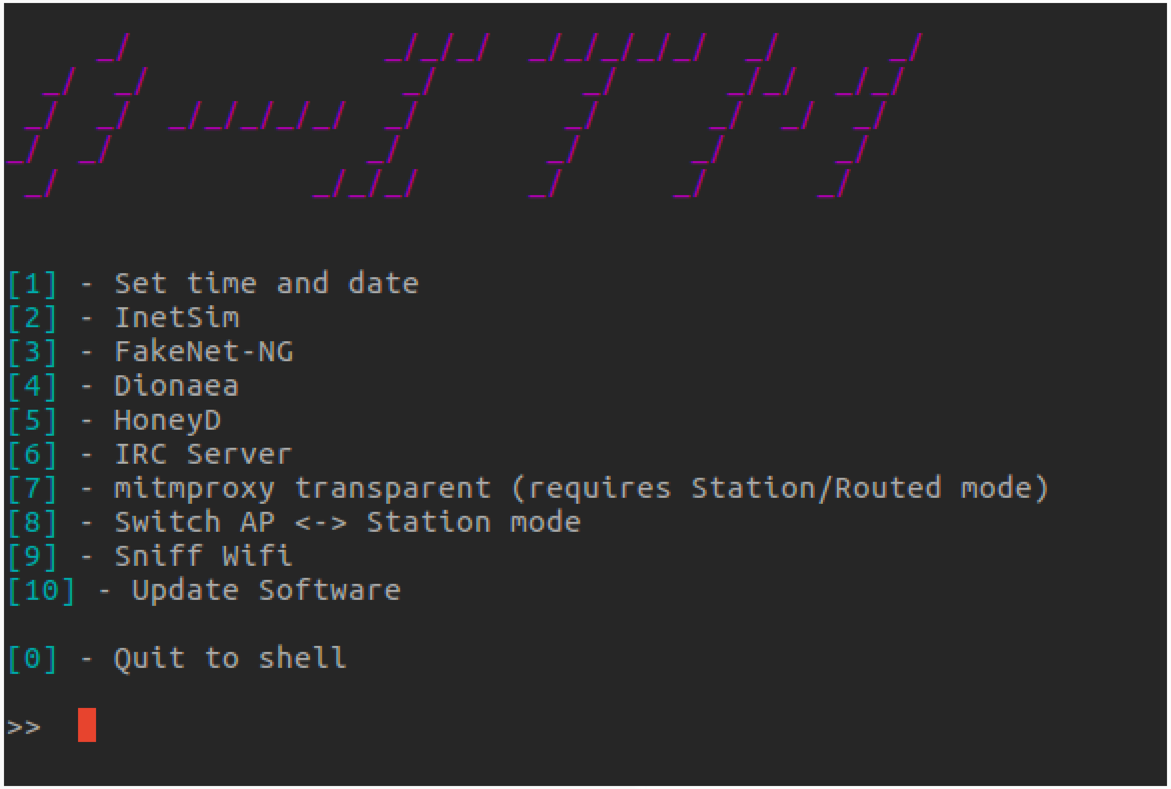
USE CASES

I'd like to expose 3 real-life use cases with their outcomes:

- IoT: reverse engineering Amazon Dash button and its communications
- Mobile: analysis of Android/iOS apps, without a lab
- Infected laptop analysis: oh my god, I have linux malware, but my sandbox does not support Linux

IOT : AMAZON DASH BUTTON

let's go in menu [8] to switch to sniff mode and then to [9]



```
     _/              _/_/_/  _/_/_/_/_/  _/      _/
  _/  _/              _/        _/      _/_/  _/_/
 _/  _/  _/_/_/_/_/  _/        _/      _/  _/  _/
_/  _/              _/        _/      _/      _/
 _/              _/_/_/      _/      _/      _/


[1] - Set time and date
[2] - InetSim
[3] - FakeNet-NG
[4] - Dionaea
[5] - HoneyD
[6] - IRC Server
[7] - mitmproxy transparent (requires Station/Routed mode)
[8] - Switch AP <-> Station mode
[9] - Sniff Wifi
[10] - Update Software

[0] - Quit to shell

>>
```

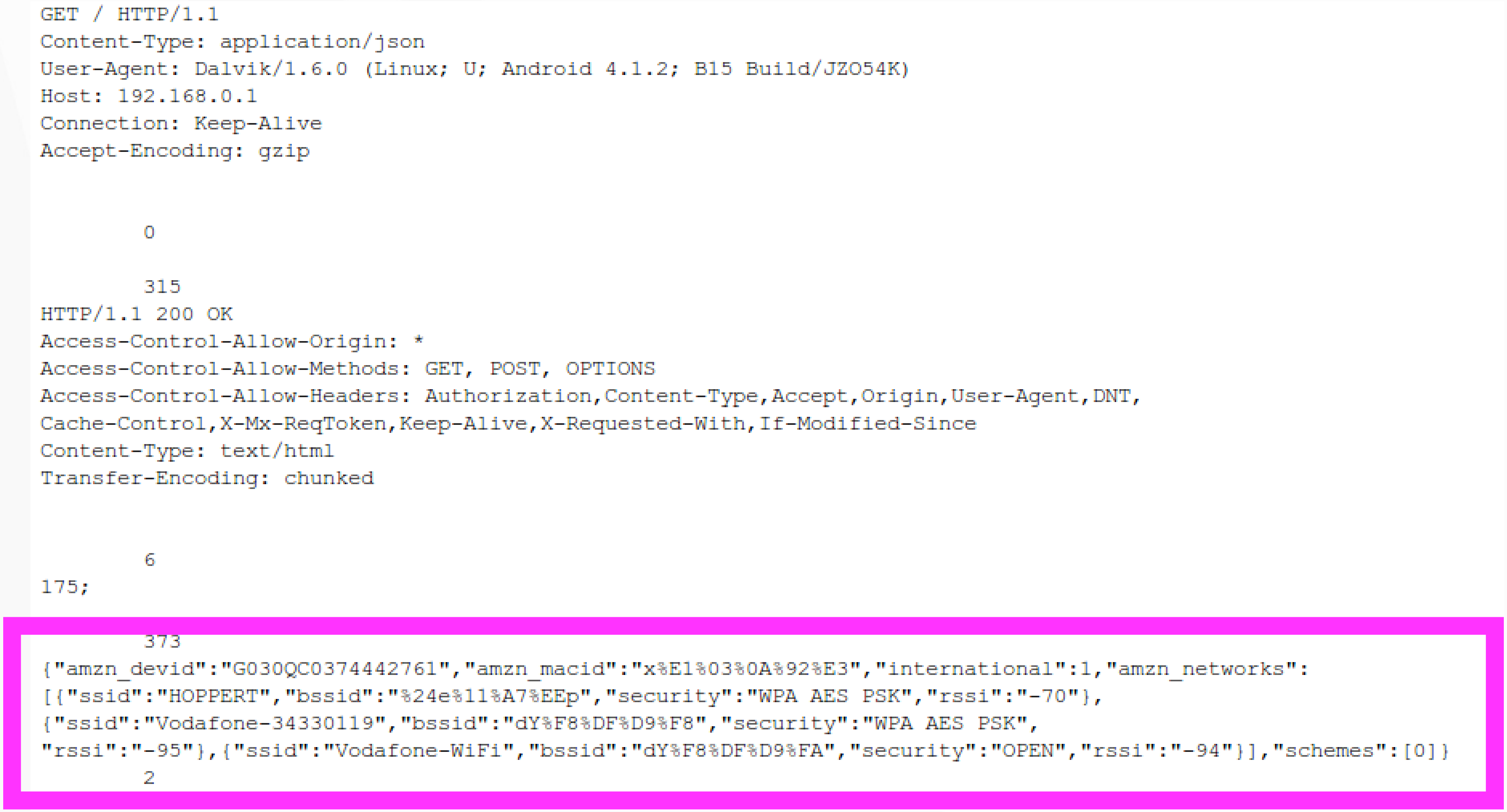
start sniff and start registering process on Amazon dash button

IOT: AMAZON DASH BUTTON

- now, let's examine the cap file:

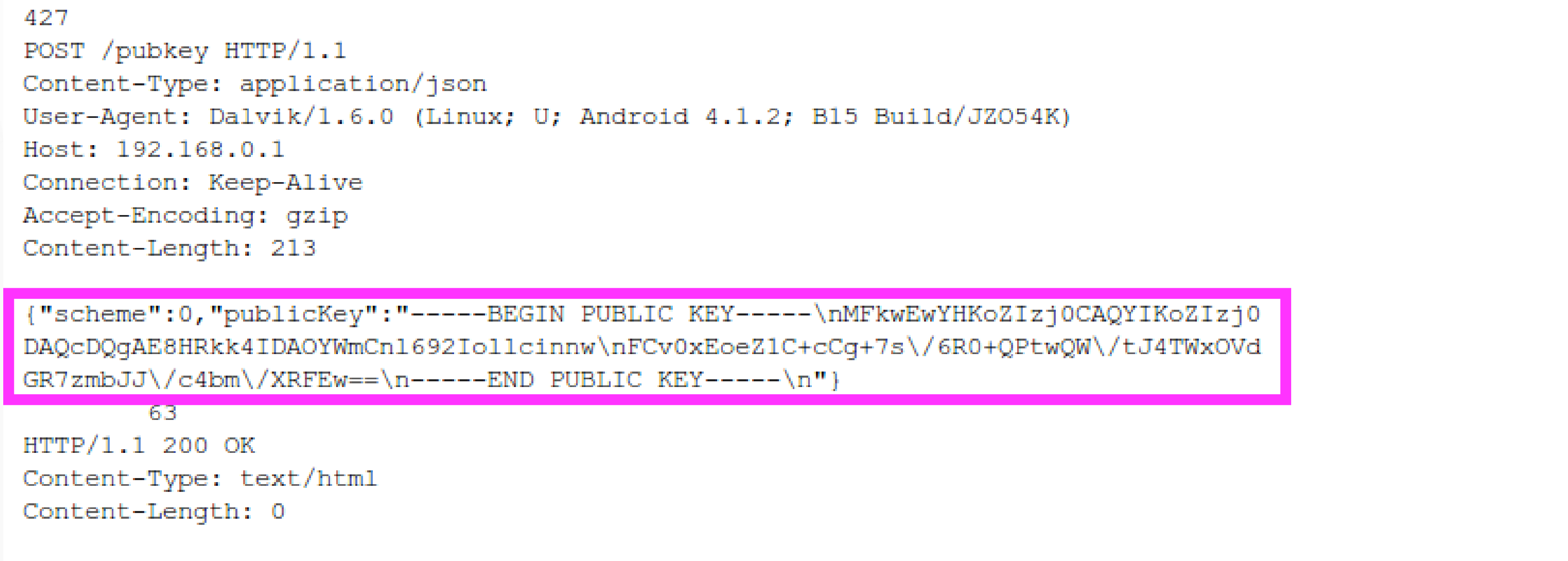
```
# tshark -r 20180121_dump.cap -q -z follow,tcp,ascii,0
```

IOT: AMAZON DASH BUTTON



```
GET / HTTP/1.1
Content-Type: application/json
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.1.2; B15 Build/JZO54K)
Host: 192.168.0.1
Connection: Keep-Alive
Accept-Encoding: gzip


        0


        315
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, OPTIONS
Access-Control-Allow-Headers: Authorization,Content-Type,Accept,Origin,User-Agent,DNT,
Cache-Control,X-Mx-ReqToken,Keep-Alive,X-Requested-With,If-Modified-Since
Content-Type: text/html
Transfer-Encoding: chunked


        6
175;

        373
{"amzn_devid":"G030QC0374442761","amzn_macid":"x%E1%03%0A%92%E3","international":1,"amzn_networks":
[{"ssid":"HOPPERT","bssid":"%24e%11%A7%EEp","security":"WPA AES PSK","rssi":"-70"},
{"ssid":"Vodafone-34330119","bssid":"dY%F8%DF%D9%F8","security":"WPA AES PSK",
"rssi":"-95"},{"ssid":"Vodafone-WiFi","bssid":"dY%F8%DF%D9%FA","security":"OPEN","rssi":"-94"}],"schemes":[0]}
        2
```

IOT: AMAZON DASH BUTTON



```
427
POST /pubkey HTTP/1.1
Content-Type: application/json
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.1.2; B15 Build/JZO54K)
Host: 192.168.0.1
Connection: Keep-Alive
Accept-Encoding: gzip
Content-Length: 213

{"scheme":0,"publicKey":"-----BEGIN PUBLIC KEY-----\nMFkwEwYHKoZIzj0CAQYIKoZIzj0
DAQcDQgAE8HRkk4IDAOYWmCnl692Iollcinnw\nFCv0xEoeZ1C+cCg+7s\/6R0+QPtwQW\/tJ4TWxOVd
GR7zmbJJ\/c4bm\/XRFEw==\n-----END PUBLIC KEY-----\n"}
        63
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 0
```

IOT : AMAZON DASH BUTTON

Let's try to intercept the traffic: go in [8] and switch to AP mode (routed or bridged)

- start mitmproxy in transparent mode with [7]
- repeat the association process, associate 0itm SSID and look what happens:

mmmmmmhh...the button tries to reach amazon services with HTTPS. Mitmproxy serves the certificate, but it's not trusted, so connection fails. OK, so we need to find a way to push a root certificate on the device. This will be our homework, but in 5 minutes we reached a good point

MOBILE: ANDROID/IOS APP

I found a very popular Android game (Subway Surfers) distributed on alternative markets. I'd like to understand if the "unofficial" versions are doing something malicious. Let's start:

- fire up O-ITM in AP mode using menu [8]. In this way it will publish an SSID named "Oitm" ready to be connected
- download the official app on your Android phone and then connect "Oitm" SSID
- go back to O-ITM and start fakenet-ng on wifi interface with menu [3]. In this way all the network connection will be intercepted by fakenet.
- NOTE: O-ITM contains a customized version of fakenet-ng with modifications to serve files specific for Android OS and let it think it is on-line. Specifically Android looks for "generate_204" url and it expects a reply "HTTP/1.0 204 No Content"
- go on the phone, remove the original App and then install the "fake" one
- starts again the fakenet-ng, go back on the phone and repeat the same actions done with the original App

MOBILE: ANDROID/IOS APP

Quick Video:

```
cesare@dell:~$
```

MOBILE: ANDROID/IOS APP

I see that my phone did this connection in both the case:

```
POST /fota/download/login.php HTTP/1.1
Content-Length: 50
Content-Type: application/x-www-form-urlencoded
Host: mota.mediatek.com
Connection: Keep-Alive

imei=3534XXXXXXXXX75&sn=1581XXXXX56&sim=&operator=
```

mmmmmh...this does not have anything to do with the App, but it's not a good new. Big Brother is always around...

MOBILE: ANDROID/IOS APP

DNS queries are the same, so it looks nothing malicious is done:

android.clients.google.com
api.vungle.com
asia.pool.ntp.org
config.inmobi.com
config.inmobi.com
connectivitycheck.android.com
connect.tapjoy.com
data.flurry.com
graph.facebook.com
hoodrunner.kiloo.com
i.w.inmobi.com
live.chartboost.com
m.facebook.com
mota.mediatek.com
mtalk.google.com
pool.ntp.org
sdktm.w.inmobi.com
subwaysurfers.kiloo-games.com
t1.hshh.org
ws.tapjoyads.com

We see that a login on facebook is tried by both the App. May be can investigate in this direction to see if something more happens...but let's move on next case

LINUX: INFECTED LAPTOP

- a user is saying his laptop is very slow...ok, let's bring our portable malware lab...and disconnect the laptop from the network
- connect the laptop to the ethernet adapter
- connect O-ITM on Wifi with our laptop...at least the "infected" laptop is isolated
- if needed, change the IP/GW of the "infected" laptop, fire up fakenet with option [3] and wait a while

Linux: Infected laptop

- ➔ OK, we are now ready to start the analysis directly on O-ITM. We can use the menu for the Fakenet log, or exit to the shell and examine the packet capture. Let's find the public IP connected by the laptop:

```
    # tshark -r packets_20180122_222828.pcap -T fields -e ip.dst |
grep -v ^192.168 | uniq

    218.211.90.199
    163.17.30.212
    163.172.229.214
```

- ➔ OK, after some "whois" I see the first IP is from Taiwan, second Taipei, the third Paris. Let's take a not on this

LINUX: INFECTED LAPTOP

- Now, let me check the connection done by the laptop to the first IP:

```
 # tshark -r packets_20180122_222828.pcap -q -z follow,tcp,ascii,0
```

```
===================================================================
Follow: tcp,ascii
Filter: tcp.stream eq 0
Node 0: 192.168.0.10:57843
Node 1: 218.211.90.199:8434
...........l......../Pentium(R) Dual-Core CPU       T4400  @ 2.20GHz............?.....\....3.13.0-129-
generic........s3
===================================================================
```

The "infected" laptop is connecting this IP on port 8434 and communicating details about the CPU and kernel version...another clue...

Linux: infected laptop

- Let me see the DNS queries now:

```
 # tshark -r packets_20180122_222828.pcap -T fields -e ip.src -e dns.qry.name -
R "dns.flags.response eq 0" -2 | sort -u

192.168.0.10    changelogs.ubuntu.com
192.168.0.10    s3.wio2lo1n3.pw
192.168.0.10    xmr.crypto-pool.fr
```

Hey, this is interesting: the first one is not so strange, the second is a bit strange, but the third…a Monero mining server. Got it! Now we know what is going on, we can start the cleanup…

WRAP UP

I exposed some real life cases where the usage of O-IMT really helped me. The distribution I'm going to release has many other tool not described here (radare2, pyew, yara rules), that can allow you to do a lot of further analysis.

I'll release an IMG file ready to be copied on a Micro SSD: everything will be preconfigured and ready to be used.

Refer to: https://github.com/cecio/

Known Limit:
The sniffer (PiO hardware) can go only with 20Mhz bandwidth

END

Thank you!